Version 3 Data Types
PART I

Practical Guide

Editor: Doug Pratt
SMS
Editor: Gunther Schadow
Regenstrief Institute for Health Care

Preface

This document is one of two parts specifying the HL7 Version 3 Data Types on an abstract layer, independent of representation.

- Part I explains Version 3 Data Types in a style that can be readily understood by interface analysts and programmers.
- Part II provides a rigorous definition of Version 3 Data Types in a style that is suited for those readers with a strong academic background in Computer Science and Mathematics.

Each part can stand on its own and is addressed to different kinds of audiences and a slightly different purpose. Both parts are normative; therefore both parts must remain consistent in their description of Version 3 Data Types. Due to its more thorough nature, Part II takes precedence over Part I in cases of conflict or unclear interpretation. Casual readers who need a quick orientation into the matter should read Part I. However, for any serious implementation work understanding the additional detail provided in Part II is usually required.

This standard is further accompanied by one or more Implementable Technology Specifications (ITS) to specify the concrete representation of the Version 3 Data Types.

Acknowledgements

This standard is the result of two years of intense work through e-mail, telephone conferences and meeting discussions. Gunther Schadow (Regenstrief Institute for Health Care) chaired this task force, and is the main author of this document. Major contributions are from Mark Tucker (Regenstrief Institute), Paul V. Biron (Kaiser Permanente), George Beeler (Mayo), and Stan Huff (Intermountain Health Care), as well as Mike Henderson (Kaiser Permanente), Anthony Julian (Mayo), Joann Larson (Kaiser Permanente), Mark Shafarman (Oacis Healthcare Sy stems), Wes Rishel (Gartner Group), and Robin Zimmerman (Kaiser Permanente). Acknowledgements for their critical review and infusion of ideas go to Bob Dolin (Kaiser Permanente), Clem McDonald (Regenstrief Institute), Kai Heitmann (HL7 Germany), Rob Seliger (Sentillion), and Harold Solbrig (Mayo). Vital support came from the members of the task force, Laticia Fitzpatrick (Kaiser Permanente), Matt Huges, Randy Marbach (Kaiser Permanente), Larry Reis (Wi zdom Systems), Carlos Sanroman (Kaiser Permanente), Greg Thomas (Kaiser Permanente). Thanks James Case (University of California, Davis), Norman Daoust (Partners HealthCare Systems), Irma Jongeneel (HL7 The Netherlands), Michio Kimura (HL7 Japan), John Molina (SMS), Richard Ohlmann (McKessonHBOC), Dawid Rowed (HL7 Australia), and Klaus Veil (Macquarie Health Corp., HL7 Australia), for sharing their expertise in critical questions. This work was made possible by the Regenstrief Institute for Health Care.

Contents

1 Introduction

What is a Data Type? Data types are the basic building blocks used to construct messages, computerized patient record documents, business objects and their transactions. Data Types define the meaning of any given field. Without knowing a field's data type, it is impossible to interpret the field's value. HL7 defines simple Data Types, such as strings and integers, as well as complex data types such as postal addresses, names, and timing specifications.

Representation of Data Values. Data values can be represented through various symbols or objects but the data value's meaning is not restricted to any particular representation. For example, the integer value "two" can be represented by the character "2," by the Roman Numeral string "II," or the binary string "0010."

This specification defines the meaning of the HL7 data types. It does **not** mandate an exact representation of data types. These representations are addressed in Implementable Technology Specifications (ITS), such as the *XML ITS for Version 3 Data Types*.

However, this specification does define so called "literal forms", i.e. concise character string representations for some of the data types. For example, this specification defines the decimal digit string "1234" as a valid literal form for some number data types. These literal forms are a

recommendation to an ITS to use this literal form if there is no such literal defined for the particular implementation technology. Note that the literal forms are defined for the data types but their literal forms do not define these data types.

Properties of Data Values. Data values have properties defined by their data type. The "fields" of "composite data types" are the most common example of such properties. However, some properties need not be represented as an explicit field if they can be inferred from some other information. What needs to be a field and what can be inferred depends on the representation, the ITS, the programming language used in implementation, etc.

Need for the Abstraction. Great pains have been taken to ensure that HL7 Version 3 is not tied to any particular implementation technology. Towards that end, this document does **not** specify the required representation of data types, nor does it specify operational implementations. As such, it is an *abstract* data type specification.

Any concrete implementation of the HL7 standards must ultimately use the built-in data types of their implementation technology. Therefore, it must be possible, indeed *simple*, to map the abstract data types of this specification to various implementation technologies. This mapping is required in order for an ITS to claim conformance to HL7 version 3.

1.1 Summary of Basic Data Types

This specification attempts to define all the data types needed for health care information interchange.

Table 1: Overview of HL7 version 3 Data Types

Name	Symbol	Description
Boolean	BL	The Boolean type stands for the values of two-valued logic. A Boolean value can be either true or false.
Encapsulated Data	ED	Data that is primarily intended for human interpretation or for further machine processing outside the scope of this specification. This includes unformatted or formatted written language, multi-media data, or structured information in as defined by a different standard (e.g., XML-signatures.) Instead of the data itself, an ED may contain only a reference (see TEL.) Note that the ST data type is a specialization of the ED data type when the ED media type is text/plain.
Character String	ST	Text data, primarily intended for machine processing (e.g., sorting, querying, indexing, etc.) Used for names, symbols, and formal expressions.) Note that the ST data type is a specialization of the ED data type when the ED media type is text/plain.
Coded Simple Value	CS	Coded data, consists of a code and display name. The code system and code system version is fixed by the context in which the CS value occurs. CS is used for coded attributes that have a single HL7-defined value set.
Coded Value	CV	Coded data, consists of a code, display name, code system, and original text. Used when a single code value must be sent.
Coded With Equivalents	CE	Coded data, consists of a coded value (CV) and, optionally, coded value(s) from other coding systems that identify the same concept. Used when alternative codes may exist.
Concept Descriptor	CD	Coded data, is like a CE with the extension of modifiers. Modifiers for codes have an optional role name and a value. Modifiers allow one to express, e.g., "FOOT, LEFT" as a postcoordinated term built from the primary code "FOOT" and the modifier "LEFT".
Instance Identifier	II	An identifier to uniquely identify an individual instance. Examples are medical record number, order number, service catalog item number, etc. Based on the ISO Object Identifier (OID)
Telecommunication Address	TEL	A telephone number or e-mail address specified as a URL. In addition, this type contains a time specification when that address is to be used, plus a code describing the kind of situations and requirements that would suggest that address to be used (e.g., work, home, pager, answering machine, etc.)
Postal Address	AD	For example, a mailing address. Typically includes street or post office Box, city, postal code, country, etc.
Entity Name	EN	A name of a person, organization, place, or thing. Can be a simple character string or may consist of several name parts that can be classified as given name, family name, nickname, suffix, etc.
Person Name	PN	A name of a person. Person names usually consist of several name parts that can be classified as given, family, nickname etc. PN is a restriction of EN.

Organization Name	ON	A name of an organization. ON name parts are typically not distinguished, but may distinguish the suffix for the legal standing of an organization (e.g. “Inc.”, “Co.”, “B.V.”, “GmbH”, etc.) from the name itself. ON is a restriction of EN.
Trivial Name	TN	A restriction of EN that is equivalent with a plain character string (ST). Typically used for the names of things, where name parts are not distinguished.
Integer Number	INT	Positive and negative whole numbers typically the results of counting and enumerating. The standard imposes no bounds on the size of integer numbers.
Real Number	REAL	Fractional numbers. Typically used whenever quantities are measured, estimated, or computed from other real numbers. The typical representation is decimal, where the number of significant decimal digits is known as the precision.
Physical Quantity	PQ	A dimensioned quantity expressing the result of measurement. It consists of a real number value and a physical unit. Physical quantities are often constrained to a certain dimension by specifying a unit representing the dimension (e.g. m, kg, s, kcal/d, etc.) However, physical quantities should not be constrained to any particular unit (e.g., should not be constrained to centimeter instead of meter or inch.)
Monetary Amount	MO	The amount of money in some currency. Consists of a value and a currency denomination (e.g., U.S.$, Pound sterling, Euro, Indian Rupee.)
Ratio	RTO	A quantity explicitly including both a numerator and a denominator (e.g. 1:128.) Only in the rare cases when the numerator and denominator must stand separate should the Ratio data type should be used. Normally, the REAL, PQ, or MO data types are more appropriate.
Point in Time	TS	A time stamp.
General Timing Specification	GTS	One or more time intervals used to specify the timing of events. Every event spans one time interval (occurrence interval). A repeating event is timed through a sequence of such occurrence intervals. Such timings are often specified not directly as a sequence of intervals but as a rule, e.g., “every other day (Mon – Fri) between 08:00 and 17:00 for 10 minutes.”

1.2 Null Values

Every data element has a “proper” value or it is considered NULL. If (and only if) it is NULL, a “null flavor” provides more detail, as shown in Table 2:

Table 2: Flavors of NULL

Concept	Symbol	Implies	Definition
no information	NI		No information whatsoever can be inferred. This is the default null flavor.
not applicable	NA	NI	No proper value is applicable in this context (e.g., last menstrual period for a male.)
unknown	UNK	NI	A proper value is applicable, but not known.
not asked	NASK	UNK	This information has not been sought (e.g., patient was not asked)
asked but unknown	ASKU	UNK	Information was sought but not found (e.g., patient was asked but didn't know)
temporarily unavailable	NAV	ASKU	Information is not available at this time but it is expected that it will be available later.
other	OTH		The actual value is not an element in the value domain of a variable. (e.g., diagnoses is not in ICD-9 code set; age exceeds 100-year upper limit restriction for this field.)
positive infinity	PINF	OTH	Positive infinity of numbers.
negative infinity	NINF	OTH	Negative infinity of numbers.
not present	*NP*		*Value is not present in a message. This is only defined in messages, never in application data! If a value is not present in a message the receiving interface must fill in that value with either an applicable default, or no-information (NI).*

Note the fine difference in coded data types between NULL/*other* on the one hand and *coded with extensibility* (CWE) on the other hand. CWE data types permit one to add values to the code set as needed, so Null/Other is almost never used. On the other hand, for CNE (coded, non-extensible) fields NULL/other is the only legal way to express a value that does not appear in the code set.

Implementable Technology Specifications do not need to represent NULL flavors in cases where their difference is insignificant to them.

2 Basic Data Types

2.1 Boolean (BL)

A Boolean value can be either "true" or "false", or, like any value, it may be NULL. The Boolean value obeys the common operators negation, conjunction, and disjunction. With the NULL value these common Boolean operations are extended as shown in the following tables:

Table 3: Truth tables for Boolean logic with NULL values

NOT	
true	*false*
false	*true*
NULL	NULL

AND	*true*	*false*	NULL
true	*true*	*false*	NULL
false	*false*	*false*	*false*
NULL	NULL	*false*	NULL

OR	*true*	*false*	NULL
true	*true*	*true*	*true*
false	*true*	*false*	NULL
NULL	*true*	NULL	NULL

2.2 Encapsulated Data (ED)

Encapsulated data can convey any data. However, in order for that data to convey meaning, encapsulated data must be decoded and further interpreted. Encapsulated data may be a plain character string, formatted text, or any of several kinds of multimedia data.

Table 4: Summary of Encapsulated Data (ED)

Name	Type	Status	Definition
BIN	BIN	mandatory	The binary data.
type	CS	mandatory	Identifies the encoding of the data and a method to interpret the data.
charset	CS	optional	Where applicable, specifies the character set and character encoding used. The charset may be implied or fixed by the ITS.
language	CS	optional	Where applicable, specifies the language of text data.
compression	CS	optional	Indicates whether the raw byte data is compressed, and what compression algorithm was used.
reference	TEL	optional	A telecommunication address that resolves to the binary data.
integrityCheck	BIN	optional	A short binary value representing a cryptographically strong checksum over the binary data.
integrityCheckAlgorithm	CS	optional	Specifies the algorithm used to compute the integrityCheck value.
thumbnail	ED	optional	An abbreviated rendition of the full data.

Encapsulated data can be present in two forms, inline or by reference. Inline data is communicated or moved as part of the encapsulated data value, whereas by-reference data may reside at a different (remote) location. The data is the same whether it is located inline or remote.

2.2.1 Binary Data (BIN)

All communicated information must ultimately be physically encapsulated as binary data. Binary data is a raw stream of bits, where a bit is identical with a Boolean value.

2.2.2 Properties of Encapsulated Data

2.2.2.1 type : CS

Identifies the encoding of the encapsulated data and identifies a method to interpret or render the data. This is specified using the MIME media types codes, defined by the *Internet Assigned Numbers Authority* (IANA).

To promote interoperability, this specification prefers certain media types to others. This is to define a greatest common denominator on which interoperability is not only possible, but that is powerful enough to support even advanced multimedia communication needs.

Table 5 below assigns a status to certain MIME media types, where the status means one of the following:

required

Every HL7 application must support at least the required media types if it supports a given kind of media. One required media-type for each kind of media exists. Some media types are required for a specific purpose, which is then indicated as "*required for …*"

recommended

Other media types are recommended for a particular purpose. For any given purpose there should be only very few additionally recommended media types and the rationale, conditions and assumptions of such recommendations must be made very clear.

indifferent

This status means, HL7 does neither forbid nor endorse the use of this media type. All media types not mentioned here by default belong into the *indifferent* category. Since there is one required and several recommended media types for most practically relevant use cases, media types of this status should be used very conservatively.

deprecated

Deprecated media types should not be used, because these media types are flawed, because there are better alternatives, or because of certain risks. Such risks could be security risks, for example, the risk that such a media type could spread computer viruses. Not every flawed media type is marked as deprecated, though. A media type that is not mentioned, and thus considered *other* by default, may well be flawed.

Table 5: Use of MIME media types

Media Type	Status	Use Case
text/plain	required **default**	For any plain text. This is the default and is equivalent to a character string (ST) data type.
text/x-hl7-ft	recommended *for compatibility*	For compatibility, this represents the HL7 v2.x FT data type. Its use is recommended only for backward compatibility with HL7 v2.x systems.
text/html	recommended	For marked-up text according to the Hypertext Mark-up Language. HTML markup is sufficient for typographically marking-up most written-text documents. HTML is platform independent and widely deployed.
application/pdf	recommended	The Portable Document Format is recommended for written text that is completely laid out and read-only. PDF is a platform independent, widely deployed, and open specification[1] with freely available creation and rendering tools.
text/sgml text/xml	indifferent	For structured character based data. There is a risk that general SGML/XML is too powerful to allow a sharing of general SGML/XML documents between different applications.
text/rtf	indifferent	The Rich Text Format is widely used to share word-processor documents.

[1] The specification is publicly available [http://partners.adobe.com/asn/developer/technotes.html#acrobat-pdf] and implemented by at least two independent parties.

		However, RTF does have compatibility problems, as it is quite dependent on the word processor. May be useful if word processor edit-able text should be shared.
application/msword	deprecated	This format is very prone to compatibility problems. If sharing of edit-able text is required, text/plain, text/html or text/rtf should be used instead.
audio/basic	required *for audio*	This is a format for single channel audio, encoded using 8bit ISDN mu-law [PCM] at a sample rate of 8000 Hz. This format is standardized by: CCITT, Fascicle III.4 –Recommendation G.711. *Pulse Code Modulation (PCM) of Voice Frequencies*. Geneva, 1972.
audio/mp3	recommended *for CD quality audio*	MPEG-1 Audio layer-3 is an audio compression algorithm and file format defined in ISO 11172-3 and ISO 13818-3. MP3 has an adjustable sampling frequency for highly compressed telephone to CD quality audio.
audio/k32adpcm	recommended *for audio compression*	ADPCM allows compressing audio data. It is defined in the Internet specification RFC 2421 [ftp://ftp.isi.edu/in-notes/rfc2421.txt]. Its implementation base is unclear.
image/png	required *for images*	Portable Network Graphics (PNG) [http://www.cdrom.com/pub/png] is a widely supported lossless image compression standard with open source code available.
image/gif	indifferent	GIF is a popular format that is universally well supported. However GIF is patent encumbered and should therefore be used with caution.
image/jpeg	required *for high color images*	This format is required for high compression of high color photographs. It is a "lossy" compression, but the difference to lossless compression is almost unnoticeable to the human vision.
image/g3fax	recommended *for FAX*	This is recommended only for fax applications.
image/tiff	indifferent	Although TIFF (Tag Image File Format) is an international standard it has many interoperability problems in practice. Too many different versions that are not handled by all software alike.
video/mpeg	required *for video*	MPEG is an international standard, widely deployed, highly efficient for high color video; open source code exists; highly interoperable.
video/x-avi	deprecated	The AVI file format is just a wrapper for many different codecs; it is a source of many interoperability problems.
model/vrml	recommended *for 3D models*	This is an openly standardized format for 3D models that can be useful for virtual reality applications such as anatomy or biochemical research (visualization of the steric structure of macromolecules)

The set of required media types, however, is very small so that no undue requirements are forced on HL7 applications, especially legacy systems. In general, no HL7 application is forced to support any given kind of media other than written text. For example, many systems just do not want to receive audio data, because those systems can only show written text to their users. It is a matter of application conformance statements to say: "I will not handle audio". Only if a system claims to handle audio media, it must support the required media type for audio.

2.2.2.2 charset : CS

For character-based encoding types, this property specifies the character set and character encoding used. The charset is defined according to Internet RFC 2278, *IANA Charset Registration Procedures*, [http://www.isi.edu/in-notes/rfc2278.txt].

The charset domain is maintained by the *Internet Assigned Numbers Authority* (IANA) [http://www.isi.edu/in-notes/iana/assignments/character-sets]. The IANA source specifies names and multiple aliases for most character sets. For the HL7's purposes, use of multiple alias names is not allowed. The standard name for HL7 is the one marked by IANA as "preferred for MIME." If IANA has not marked one of the aliases as "preferred for MIME" the main name shall be the one used for HL7.

Table 6 lists a few of the IANA defined character sets that are of interest to current HL7 members. The definitions of the "status" column is as given for Table 5.

Table 6: Select Character Set Codes as defined by IANA

Code	Status	Description
US-ASCII	required	ANSI X3.4-1968
UTF-8	required	8 bit Unicode Transfer Format [RFC 2279]. This is the default character set

	for Unicode	(ISO 10646/Unicode) and encoding for XML and natively supported by Java. It is backward compatible to 7-bit US-ASCII.
ISO-10646-UCS-2	deprecated	Unicode ISO 10646, the 16 bit per character Basic Multilingual Plane. Unicode has a special protocol to specify the byte order, which must be followed. To avoid byte ordering problems (and – for the western part of the world – to conserve bandwidth) the UTF-8 encoding should be used.
ISO-10646-UCS-4	deprecated	Unicode ISO 10646, the full code-set (32-bit per character.) Unicode has a special protocol to specify the byte order, which must be followed. To avoid byte ordering problems (and – for the western part of the world – to conserve bandwidth) the UTF-8 encoding should be used.
UTF-7	indifferent	7 bit Unicode Transfer Format [RFC 2152]. This is a Unicode encoding that is sure to be safe for older communication links or file formats that remove the 7th bit of each transferred byte.
ISO-8859-1	indifferent	ISO 8859 Latin-1 character set is native on western European (and U.S.) Microsoft Windows installations and on many Unix/X-Windows systems.
ISO-8859-2	indifferent	ISO 8859 Latin-2 character set for the Slavic languages of Central Europe (Polish, Czech).
ISO-8859-5	indifferent	ISO 8859 Cyrillic character set for the languages Russian, Bulgarian, Byelorussian, Macedonian, Serbian and Ukrainian.
JIS-2022-JP	indifferent	ISO 2022 is a character-encoding framework in which multilingual code-pages can be switched in and out. JIS-2022-JP, is ISO 2022 as released as a Japanese Information Standard and as the Internet specification *Japanese Character Encoding for Internet Messages* [RFC 1468].
EBCDIC	indifferent	Extended binary-coded decimal interchange code. A coded character set of 256 8-bit characters commonly used by IBM mainframes.

2.2.2.3 language : CS

For character based information the language property specifies the language of the text. The HL7 table for human languages is based on RFC 1766, *Tags for the Identification of Languages* [http://www.isi.edu/in-notes/rfc1766.txt]. It is a set of pre-coordinated pairs of one 2-letter ISO 639 language code and one 2-letter ISO 3166 country code (e.g., en-us [English, United States]).

Language tags do not modify the meaning of the characters found in the text; they are only an advice on if and how to present or communicate the text. For this reason, any system or site that does not deal with multilingual text or names in the real world can safely ignore the language property.

2.2.2.4 compression : CS

Indicates whether the raw byte data is compressed, and what compression algorithm was used.

Table 7: Compression Algorithms

Name	Code	Status	Description and Comment
deflate	DF	required	The "deflate" compressed data format as specified in RFC 1951 [ftp://ftp.isi.edu/in-notes/rfc1951.txt].
Gzip	GZ	other	A compressed data format that is compatible with the widely used GZIP utility as specified in RFC 1952 [ftp://ftp.isi.edu/in-notes/rfc1952.txt] (uses the *deflate* algorithm.)
Zlib	ZL	other	A compressed data format that also uses the *deflate* algorithm. Specified as RFC 1950 [ftp://ftp.isi.edu/in-notes/rfc1950.txt]
compress	Z	deprecated	Original UNIX compress algorithm and file format using the LZC algorithm (a variant of LZW). Patent encumbered and less efficient than *deflate*.

2.2.2.5 reference : TEL

A telecommunication address (TEL), such as a URL for HTTP or FTP, which will resolve to precisely the same binary data that could as well have been provided as inline data.

The semantic value of an encapsulated data value is the same, regardless whether the data is present inline data or just by-reference. However, an encapsulated data value without inline data behaves differently, since any attempt to examine the data requires the data to be downloaded from the reference.

An encapsulated data value may have both inline data and a reference. The reference must point to the same data as provided inline.

2.2.2.6 integrityCheck : BIN

The integrity check is a short binary value representing a cryptographically strong checksum that is calculated over the binary data. The purpose of this property, when communicated with a reference is for anyone to validate later whether the reference still resolved to the same data that the reference resolved to when the encapsulated data value with reference was created.

The integrity check is calculated according to the integrity check algorithm. By default, the *Secure Hash Algorithm-1* (SHA-1) shall be used. The integrity check is binary encoded according to the rules of the integrity check algorithm.

The integrity check is calculated over the raw binary data that is contained in the data component, or that is accessible through the reference. No transformations are made before the integrity check is calculated. If the data is compressed, the Integrity Check is calculated over the compressed data.

2.2.2.7 integrityCheckAlgorithm : CS

Specifies the algorithm used to compute the integrityCheck value.

Table 8: Integrity Check Algorithm

Name	Code	Description
Secure Hash Algorithm – 1	SHA-1	This algorithm is defined in FIPS PUB 180-1: *Secure Hash Standard*. As of April 17, 1995.

2.2.2.8 thumbnail : ED

A thumbnail is an abbreviated rendition of the full data A thumbnail requires significantly fewer resources than the full data, while still maintaining some distinctive similarity with the full data. A thumbnail is typically used with by-reference encapsulated data. It allows a user to select data more efficiently before actually downloading through the reference.

For example, a large image may be represented by a small image; a high quality audio sequence by a shorter low-quality audio; a movie may be represented by a shorter clip; text may be summarized to an abstract.

A thumbnail may not itself contain a thumbnail.

2.3 Character String (ST)

Character string is a restricted encapsulated data type (ED), whose type property is fixed to *text/plain*, and whose data must be inlined and not compressed. Thus, the properties compression, reference, integrity check, algorithm, and thumbnail are not applicable. The character string data type is used when the appearance of text does not bear meaning, which is true for formalized text and all kinds of names.

Table 9: Summary of Character String (ST)

Name	Type	Status	Definition
data	BIN	mandatory	The binary data of the character string.
charset	CS	optional	Specifies the character set and character encoding used.
language	CS	optional	Specifies the language of text data.

The character string (ST) data type interprets the encapsulated data as character data (as opposed to bits), depending on the charset property. In other words, the string S1 "Rose" is equal to the string S2 "Rose" even if S1 is ASCII-encoded (hex '526f7365') and S2 is EBCDIC-encoded (hex 'd996a285').

A character string must at least have one character or else it is NULL. The length of a string is the number of characters, not the number of encoded bytes. Byte encoding is an ITS issue and is not relevant on the application layer.

2.4 Concept Descriptor (CD)

A concept descriptor represents any kind of "concept." A concept is a class of "things." Concepts are frequently represented by codes. Diagnosis, procedures, and specialties are examples of concepts. The CD refers to a concept usually by citing a code defined in a coding system.

A major distinction exists between codes and identifiers. Codes are values that stand for classes or properties of things (sometimes referred to as *universals*). For example, "F150" is a code referring to a class of Ford pickup trucks. There are thousands of F150's on the road, all of which are classified as full-size Ford pickup trucks.

Identifiers on the other hand refer to individual things. Every Ford F150 has a unique Vehicle Identification Number (VIN), an identifier that refers to an individual truck.

Diagnosis codes, procedure codes, medication codes, gender, marital status and religion codes are examples of, as their name implies, codes. Medical record numbers, Social Security Numbers, Provider IDs, and Manufacturer IDs are examples of identifiers.

As we will see, there are different ways to represent codes and identifiers, depending on the situation. The following attributes may be used to represent a code:

Table 10: Summary of Concept Descriptor (CD)

Name	Description
code	A string containing the value of the code (e.g., "F150")
displayName	A string containing a short, human-readable description of the code. ("Ford F150 Full-size Pickup Truck")
codeSystem	An Object Identifier (OID) that uniquely identifies the code system to which the code belongs (e.g., "106.75.314.67.89.24," where this uniquely identifies Ford Motor Company's set of model numbers).
codeSystemName	A string containing a short, human-readable description of the code system (e.g., "Ford Car and Truck Models ").
codeSystemVersion	A string qualifying the version of the code system (e.g., "Model Year 2001").
originalText	This is the text, phrase, etc., that is the basis for the coding. (e.g., "The new truck purchased for hospital facility maintenance was a Ford model F150 …").
modifier	Some code systems permit modifiers, additional codes that refine the meaning represented by the primary code. HL7 Version 3 accommodates a list of modifiers. Continuing with our truck example, the list of modifiers "Body-ECAB, Eng-V8, EM-CE" modify "F150" to designate that the truck has an extended cab, V8 engine, and California Emissions package. "Body-," "Eng-," and "EM" designate the roles (body, engine, emissions) represented by the codes "ECAB," "V8," and "CE."
translation	Quite often in an interfaced environment, codes need to be translated into one or more other coding systems. In our example, the California DMV may have their own code

Some code systems define certain style options to their code values. For example, the U.S. National Drug Code (NDC) has a dash and a non-dash form. An example for the dash form may be 1234-5678-90 when the non-dash form is 01234567890. Another example for this problem is when certain ISO or ANSI code tables define optional alphanumeric and numeric forms of two or three character lengths all in one standard.

In the case where code systems provide for multiple representations, HL7 shall make a ruling about which is the preferred form. HL7 shall document that ruling where that respective external coding system is recognized. HL7 shall decide upon the preferred form based on criteria of practicality and

common use. In absence of clear criteria of practicality and common use, the safest, most extensible, and least stylized (the least decorated) form shall be given preference.

The CD data type is the basis for each of the "coded" types in this section. The CD itself is detailed in Part II of this specification, but as it is rarely expected to be used directly in a message or document, we just mention it in this part of the document.

2.4.1 Coded Simple Value (CS)

The Coded Simple Value (CS) is the simplest representation of a code is only explicitly mentions the properties listed in Table 11.

Table 11: Summary of Coded Simple Value (CS)

Name	Type	Status
code	ST	mandatory
displayName	ST	optional

The code system and code system version are fixed by the context in which the CS value occurs. Original text is not applicable to CS values.

For CS values, the designation of the domain qualifier will always be CNE (*coded, non-extensible*) and the context determines unambiguously which HL7 value set applies.

2.4.2 Coded Value (CV)

The Coded Value (CV) data type only explicitly mentions the properties listed in Table 12.

Table 12: Summary of Coded Value (CV)

Name	Type	Status
code	ST	mandatory
displayName	ST	optional
codeSystem	OID	mandatory
codeSystemName	ST	optional
codeSystemVersion	ST	optional
originalText	ST	optional

This type is used when any reasonable use case will require only a single code value to be sent. It cannot be used in circumstances where multiple alternative codes for a given value are desired.

2.4.3 Coded With Equivalents (CE)

The Coded with Equivalents (CE) data type only explicitly mentions the properties listed in Table 13.

Table 13: Summary of Coded with Equivalents (CE)

Name	Type	Status
code	ST	mandatory
displayName	ST	auxilary
codeSystem	OID	mandatory
codeSystemName	ST	auxilary
codeSystemVersion	ST	optional
originalText	ED	auxilary
translation	SET<CV>	optional

The CE type is used when the use case indicates that alternative codes may exist and where it is useful to communicate these. The CE type provides for a primary code value, plus a set of alternative or equivalent representations.

2.4.4 Coded With Category (CC)

The data type "Coded with Category" (CC) is a specific profile of the concept descriptor (CD) used for certain coded attributes, where a coarse-grained category code defined by HL7 is communicated with a fine grained externally or locally defined code.

Table 14: Summary of Primary Properties of Coded with Category (CC)

Name	Type	Status	Definition
code	ST	mandatory	The plain code symbol
displayName	ST	optional	A name or title for the code, under which the sending system shows the code value to its users
codeSystem	OID	mandatory	Specifies the code system that defines the code
codeSystemName	ST	optional	A common name of the coding system
codeSystemVersion	ST	optional	If applicable, a version descriptor defined specifically for the given code system
originalText	ED	optional	The text or phrase used as the basis for the coding
modifier		conditional	For one modifier that tells the HL7-defined category of the coded concept. Only if the codeSystem is not registered with HL7
name	CS	fixed	Fixed to "has-generalization" (GEN).
value	CS	mandatory	An HL7 defined code for the category of the concept.
translation	SET<CV>	optional	A set of other concept descriptors that translate this concept descriptor into other code systems.

2.5 Instance Identifier (II)

The Instance Identifier (II) data type is used to uniquely identify an instance, thing or object. Examples are object identifier for HL7 RIM objects, medical record number, order id, service catalog item id, Vehicle Identification Number (VIN), etc. Instance identifiers are defined based on ISO object identifiers.

Table 15: Summary of Instance Identifier (II)

Name	Type	Status	Definition
extension	ST	optional	The value of the identifier, unique within its assigning authority's namespace.
root	OID	mandatory	A unique identifier that guarantees the global uniqueness of the instance identifier. The root alone may be the entire instance identifier, an extension value is not needed.
assigningAuthorityName		optional	A human readable name or mnemonic for the assigning authority. This name is provided solely for the convenience of unaided humans interpreting an II value. Note: no automated processing must depend on the assigning authority name to be present in any form.
validTime	IVL<TS>	optional	If applicable, specifies during what time the identifier is valid. By default, the identifier is valid indefinitely. Any specific interval may be undefined on either side indicating unknown effective or expiry time. Note: identifiers for information objects in computer systems should not have restricted valid times, but should be globally unique at all times. The identifier valid time is provided mainly for real-world

identifiers, whose maintenance policy may include expiry (e.g., credit card numbers.)

2.5.1 ISO Object Identifier (OID)

An ISO Object Identifier (OID) is a globally unique string consisting of numbers and dots (e.g., 2.16.840.1.113883.3.1). This string expresses a tree data structure, with the left-most number representing the root and the right-most number representing a leaf.

Each branch under the root corresponds to an assigning authority. Each of these assigning authorities may, in turn, designate its own set of assigning authorities that work under its auspices, and so on down the line. Eventually, one of these authorities assigns a unique (to it as an assigning authority) number that corresponds to a leaf node on the tree. The leaf may represent an assigning authority (in which case the OID identifies the authority), or an instance of an object. An assigning authority owns a namespace, consisting of its sub-tree.

HL7 shall establish an OID registry and assign OIDs in its branch for HL7 users and vendors upon their request. HL7 shall also assign OIDs to public identifier-assigning authorities both U.S. nationally (e.g., the U.S. State driver license bureaus, U.S. Social Security Administration, HIPAA Provider ID registry, etc.) and internationally (e.g., other countries Social Security Administrations, Citizen ID registries, etc.) The HL7 assigned OIDs must be used for these organizations, regardless whether these organizations have other OIDs assigned from other sources.

2.5.2 Further Considerations

Some identifier schemes define certain style options to their code values. For example, the U.S. Social Security Number (SSN) is normally written with dashes that group the digits into a pattern "123-12-1234". However, the dashes are not meaningful and a SSN can just as well be represented as "123121234" without the dashes.

In the case where identifier schemes provide for multiple representations, HL7 shall make a ruling about which is the preferred form. HL7 shall document that ruling where that respective external identifier scheme is recognized. HL7 shall decide upon the preferred form based on criteria of practicality and common use. In absence of clear criteria of practicality and common use, the safest, most extensible, and least stylized (the least decorated) form shall be given preference.

HL7 may also decide to map common external identifiers to the value portion of the II.root OID. For example, the U.S. SSN could be represented as 2.16.840.1.113883.4.1.123121234. The criteria of practicality and common use will guide HL7's decision on each individual case.

2.6 Telecommunication Address (TEL)

A telecommunication address is a locator for some resource (information or services) mediated by telecommunication equipment. The semantics of a telecommunication address is that a communication entity responds to that address (the responder) and therefore can be contacted by a communication initiator. A basic example is a telephone number; dial (734) 677-777, a telephone at HL7 headquarters will ring, and an HL7 associate (or voice mail) will respond.

The responder of a telecommunication address may be an automatic service that can respond with information (e.g., FTP or HTTP services; a telephone answering machine.) In such case a telecommunication address is a reference to that information accessible through that address. A telecommunication address value can thus be resolved to some information (in the form of encapsulated data, ED.)

A given telecommunication address value may have limited validity through time and may be tagged by a use code to indicate under what circumstances a specific telecommunication address may be preferred among a set of alternatives.

Table 16: Summary of Telecommunication Address (TEL)

Name	Type	Status	Definition
URL	URL	mandatory	The essence of a telecommunication address is a Universal Resource Locator.
use	SET<CS>	optional	A code advising a system or user which telecommunication address in a set of like addresses to select for a given telecommunication need.
validTime	GTS	optional	Identifies the periods of time during which the telecommunication address can be used. For a telephone number, this can indicate the time of day in which the party can be reached on that telephone. For a web address, it may specify a time range in which the web content is promised to be available under the given address.

2.6.1 Universal Resource Locator (URL)

A Universal Resource Locator (URL) is a type of telecommunications address specified as Internet standard RFC 1738 [http://www.isi.edu/in-notes/rfc1738.txt]. The URL specifies the protocol and the contact point defined by that protocol for the resource. Notable uses of the telecommunication address data type is for telephone and telefax numbers, e-mail addresses, Hypertext references, FTP references, etc.

URLs are normally represented in a character string, formatted as "<scheme>:<address>," where the most common schemes are:

Table 17: URL Schemes

Code	Status	Definition
tel	required	A voice telephone number [draft-antti-telephony-url-11.txt].
fax	required	A telephone number served by a fax device [draft-antti-telephony-url-11.txt].
mailto	required	Electronic mail address [RFC 2368].
http	required	Hypertext Transfer Protocol [RFC 2068].
ftp	required	The File Transfer Protocol (FTP) [RFC 1738].
file	deprecated	Host-specific local file names [RCF 1738]. Note that the file scheme works only for local files. There is little use for exchanging local file names between systems, since the receiving system likely will not be able to access the file.
telnet	other	Reference to interactive sessions [RFC 1738]. Some sites, (e.g., laboratories) have TTY based remote query sessions that can be accessed through telnet.
modem	other	A telephone number served by a modem device [draft-antti-telephony-url-11.txt].

The address portion of the URL is a character string whose format is entirely defined by the URL scheme.

Telephone and FAX Numbers. There is no special data type for telephone numbers. Telephone numbers are telecommunication addresses and are specified as a URL. The voice telephone URLs begin with "tel:" and fax URLs begin with "fax:"

The telephone number URL is defined in the Internet RFC 2806 [http://www.isi.edu/in-notes/rfc2806.txt] *URLs for Telephone Calls*. For examples, "`tel:+1(317)630-7960`" is a phone number, and "`fax:+49(30)8101-724`" is a FAX number. The global absolute telephone numbers starting with the "+" and country code are preferred. Separator characters serve as decoration but have no meaning for the telephone number, thus "`tel:+13176307960`" and "`fax:+49308101724`" are the same telephone and FAX numbers as the previous respective examples.

2.6.2 The "use code" helps selecting among alternatives

The telecommunication use code's purpose is to suggest or discourage the use of a particular telecommunication address; it is not a complete classification for equipment types or locations.

Table 18: Telecommunication Address Use Code

Concept	Code	Implies	Definition
home	H		A communication address at a home, attempted contacts for business purposes might intrude privacy and chances are one will contact family or other household members instead of the person one wishes to call. Typically used with urgent cases, or if no other contacts are available.
primary home	HP	H	The primary home, to reach a person after business hours.
vacation home	HV	H	A vacation home, to reach a person while on vacation.
work place	WP		An office address. First choice for business related contacts during business hours.
answering service	AS		An automated answering machine used for less urgent cases and if the main purpose of contact is to leave a message or access an automated announcement.
emergency contact	EC		A contact specifically designated to be used for emergencies. This is the first choice in emergencies, independent of any other use codes.
pager	PG		A paging device suitable to solicit a callback or to leave a very short message.
mobile contact	MC		A telecommunication device that moves and stays with its owner. May have characteristics of all other use codes, suitable for urgent matters, not the first choice for routine business.

2.7 Postal Address (AD)

The postal address data type is used to communicate mailing and home or office addresses. The main use of such data is to allow printing mail labels, or to allow a person to physically visit that address.

The postal address data type is not supposed to be a container for additional information that might be useful for finding geographic locations (e.g., GPS coordinates) or for performing epidemiological studies. Only those parts of addresses that are conventional for designating mailboxes or home or office addresses are part of the address data type. HL7 has other and better ways to handle global positioning or census units.

The postal address data type is essentially a sequence of address part values. There are exact rules that govern the formatting of addresses on labels. These rules are detailed in Part II of this specification.

2.7.1 Address Part (ADXP)

An address part is essentially a character string that may have a type-tag signifying its role in the address. Typical parts that exist in about every address are street, house number, or post box, ZIP code, city, country but other roles may be defined regionally, nationally, or on an enterprise level (e.g. in military addresses). Addresses are usually broken up into lines, which is indicated by special line-break tokens (e.g., carriage return).

Table 19: Summary of Address Part (ADXP)

Name	Type	Status	Definition
ST	ST	mandatory	The address part data
type	CS	optional	Indicates whether an address part is the street, city, country, postal code, post box, etc.

Addresses are conceptualized as text with added mark-up. The mark-up may break the address into lines and may describe in detail the role of each address part if it is known. Address parts occur in the address in the order in which they would be printed on a mailing label. The model is similar to HTML or XML markup of text.

2.7.1.1 type : CS

Indicates whether an address part is the street, city, country, postal code, post box, etc. If the type is NULL the address part is unclassified and simply appears on the label as is.

Table 20: Address Part Type Code

Concept	Code	Definition
delimiter	DEL	Delimiters are printed without framing white space. If no value component is provided, the delimiter appears as a line break.
country	CNT	Country
state or province	STA	A sub-unit of a country with limited sovereignty in a federally organized country.
city	CTY	City
postal code	ZIP	A postal code designating a region defined by the postal service.
street name	STR	Street name or number.
house number	HNR	The number of a house or lot alongside the street. Also known as "primary street number", but does not number the street but the house.
direction	DIR	direction (e.g., N, S, W, E)
additional locator	ADL	This can be a unit designator, such as apartment number, suite number, or floor. There may be several unit designators in an address (e.g., “3rd floor, Appt. 342”.) This can also be a designator pointing away from the location, rather than specifying a smaller location within some larger one (e.g., Dutch "t.o." means "opposite to" for house boats located across the street facing houses.)
post box	POB	A numbered box located in a post station.

2.7.2 Properties of Postal Address

Addresses are essentially sequences of address parts, but add a “use” code and a valid time range for information about if and when the address can be used for a given purpose.

Table 21: Summary of Postal Address (AD)

Name	Type	Status	Default	Constraint	Definition
	LIST<ADXP>	mandatory	NULL		The address data
use	SET<CS>	optional	NULL	AddressUse	A code advising a system or user which address in a set of like addresses to select for a given purpose
validTime	GTS	optional	NULL		Identifies the periods of time during which the address can be used. Typically used to refer to different addresses for different times of the year or to refer to historical addresses.

An address without specific use code might be a default address useful for any purpose, but an address with a specific use code would be preferred for that respective purpose.

Table 22: Address Use Code

Concept	Code	Implies	Definition
visit address	RES		Used primarily to visit an address.
mail address	PST		Used to send mail.
invoice address	INV	PST	An address at which to send invoices
temporary address	TMP		A temporary address, may be good for visit or mailing. Note that an address history can provide more detailed information.
bad address	BAD		A flag indicating that the address is bad, in fact, useless.
home	H		A private (home) address.
primary home	HP	H	The primary home.
vacation home	HV	H	A vacation home, to reach a person while on vacation.
work place	WP		An office address.

2.7.3 Examples

The following are examples of addresses in an XML encoded form, where the XML tag is the address part role and the data content is the address part value. The use of XML in these examples does not preempt any XML implementation technology specification, it is solely for the purpose of this example.

1050 Wishard Blvd. RG 5th floor,
Indianapoli, IN 46240.

has the following three valid encodings

```
<AD purpose="RES">
   1050 Wishard Blvd, RG 5th floor<DEL/>
   Indianapolis, IN 46240
</AD>
```

```
<AD purpose="RES">
   <STR>1050 Wishard Blvd</STR><ADL>RG 5th floor</ADL><DEL/>
   <CTY>Indianapolis</CTY><STA>IN</STA><ZIP>46240</ZIP>
</AD>
```

```
<AD purpose="RES">
   <HNR>1050</HNR><STR>Wishard Blvd</STR><ADL>RG 5th
floor</ADL><DEL/>
   <CTY>Indianapolis</CTY><STA>IN</STA><ZIP>46240</ZIP>
</AD>
```

the second encoding in this example is more specific about the role of the address parts than the first one. The third is even more specific. The first form would result from a system that only stores addresses as line 1, line 2, etc. The second form is the typical form seen in the U.S., where street address is sometimes separated, and city, state and ZIP code are always separated. However, in the U.S. the house number is not usually separated from the street address, where in Germany many systems keep house number as separate fields (third example.)

This example shows the strength of the mark-up approach to addresses. A typical German system that stores house number and street name in separate fields would print the address with street name first followed by the house number. For U.S. addresses, this would be wrong as the house number in the U.S. is written before the street name. The marked-up address allows keeping the natural order of address parts and still understanding their role.

2.8 Entity Name (EN)

An entity name data value specifies a name of a person, organization, place or thing. Examples for entity name values are “Jim Bob Walton, Jr.”, “Health Level Seven, Inc.”, “Lake Tahoe”, etc. An entity name may be as simple as a character string or may consist of several entity name parts (ENXP), such as, “Jim”, “Bob”, “Walton”, and “Jr.”, “Health Level Seven” and “Inc.”, “Lake” and “Tahoe”.

The entity name data type is essentially a sequence of entity name part values. There are exact rules that govern the formatting of names on labels, badges, etc. These rules are detailed in Part II of this specification.

2.8.1 Entity Name Part (PNXP)

An entity name part is a character string token that may have a type code signifying the role of the part in the whole entity name. Typical name parts that exist in about every name are given names, and family names, titles, etc.

Table 23: Summary of Entity Name Part (ENXP)

Name	Type	Status	Default	Constraint	Definition
	ST	mandatory	NULL		The entity name part data
type	CS	optional	NULL	EntityNamePartType	Indicates whether the name part is a given name, family name, prefix, suffix, etc.
qualifier	SET<CS>	optional	NULL	EntityNameQualifier	A set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type

Each name part may have a type code, identifying given names, family names, prefix, suffix, etc. The type code may not be available for unknown person names.

Table 24: Name Part Type

Name	Code	Definition
family	FAM	Family name, this is the name that links to the genealogy. In some cultures (e.g. Eritrea) the family name of a son is the first name of his father.
given	GIV	Given name (don't call it "first name" since this given names do not always come first)
	MID	
prefix	PFX	A prefix has a strong association to the immediately following name part. A prefix has no implicit trailing white space (it has implicit leading white space though). Note that prefixes can be inverted.
suffix	SFX	A suffix has a strong association to the immediately preceding name part. A prefix has no implicit leading white space (it has implicit trailing white space though). Suffices can not be inverted.
delimiter	DEL	A delimiter has no meaning other than being literally printed in this name representation. A delimiter has no implicit leading and trailing white space.

Each name part may have a qualifier. The qualifier is a set of codes each of which specifies a certain subcategory of the name part in addition to the main name part type. For example, a given name may be flagged as a nickname, a family name may be a pseudonym or a name of public records.

Table 25: Name Part Qualifier

Name	Code	Definition
Name change classifiers describe how a name part came about. More than one value allowed.		
birth	BR	A name that a person had shortly after being born. Usually for family names but may be used to mark given names at birth that may have changed later.
unmarried	MD	A name that a person (either sex) had immediately before her/his first marriage. Usually called "maiden name", this concept of maiden name is only for compatibility with cultures that keep up this traditional concept. In most cases maiden name is equal to birth name. If there are adoption or deed polls before first marriage the maiden name should specify the last family name a person acquired before giving it up again through marriage.
chosen	CH	A name that a person assumed because of free choice. Most systems may not track this, but some might. Subsumed in the concept of "chosen" are pseudonym (alias), and *deed poll*. The difference in civil dignity of the name part is given through the R classifier below. I.e. a deed poll creates a chosen name of record, whereas a pseudonym creates a name not noted in civil records.
adoption	AD	A name that a person took on because of being adopted. Adoptions may happen for adults too and may happen after marriage. Whether adoption name or the birth name is considered the "maiden" name is not fully defined and may, as always, simple depend on the discretion of the person or a data entry clerk.
spouse	SP	The name assumed from the partner in a marital relationship (hence the "M"). Usually the spouse's family name. Note that no inference about gender can be made from the existence of spouse names.
Affix types. Usually only one value per affix.		
voorvoegsel	VV	A Dutch "voorvoegsel" is something like "van" or "de" that might have indicated nobility in the past but no longer so. Similar prefixes exist in other languages such as Spanish, French or Portugese.
academic	AC	Indicates that a prefix like “Dr.” or a suffix like “M.D.” or “Ph.D.” is an academic title.
professional	PR	Primarily in the British Imperial culture people tend to have an abbreviation of their professional organization as part of their credential suffices.
nobility	NB	In Europe and Asia, there are still people with nobility titles (aristocrats.) German "von" is generally a nobility title, not a mere voorvoegsel. Others are "Earl of" or "His Majesty King of..." etc. Rarely used nowadays, but some systems do keep track of this.
legal status	LS	For organizations a suffix indicating the legal status, e.g., “Inc.”, “Co.”, “AG”, “GmbH”, “B.V.” “S.A.”, “Ltd.” etc.
Additional qualifiers. More than one value allowed.		
nick	NK	Indicates that the name part is a nickname. Not explicitly used for prefixes and suffixes, since those inherit this flag from their associated significant name parts. Note that most nicknames are given names although it is not required.
callme	CL	A callme name is (usually a given name) that is preferred when a person is directly addressed.

record	RE	This flag indicates that the name part is known in some official record. Usually the antonym of nickname. Note that the name purpose code "license" applies to all name parts or a name, whereas this code applies only to name name part.
initial	IN	Indicates that a name part is just an initial. Initials do not imply a trailing period since this would not work with non-Latin scripts. Initials may consist of more than one letter, e.g., "Ph." could stand for "Philippe" or "Th." for "Thomas".
weak	WK	Used only for prefixes and suffixes (affixes). A weak affix has a weaker association to its main name part than a genuine (strong) affix. Weak prefixes are not normally inverted. When a weak affix and a strong affix occur together, the strong affix is closer to its associated main name part than the weak affix.
invisible	HD	Indicates that a name part is not normally shown. For instance, traditional maiden names are not normally shown. "Middle names" may be invisible too.

Entity names have no additional properties that add information to the sequence of person name parts.

2.8.2 Examples

The following shows examples of entity names in an XML encoded form, where the XML tag is the entity name part type and the data content is the entity name part value. The use of XML in these examples does not preempt any XML implementation technology specification; it is solely for the purpose of this example.

A very simple encoding of "John W. Doe" would be:

```
<EN>
   <GIV>John</GIV>
   <GIV>W.</GIV>
   <FAM>Doe</FAM>
</EN>
```

none of the special qualifiers need to be mentioned if they are unknown or irrelevant. The next example shows extensive use of multiple given names, prefixes, suffixes, for academic degrees, nobility titles, *vorvoegsels* ("van"), and professional designations.

```
<EN>
   <PFX Q="AC">Dr. phil.  </PFX>
   <GIV>Regina</GIV><GIV>Johanna</GIV><GIV>Maria</GIV>
   <PFX Q="NB">Gräfin_</PFX><PFX Q="VV">von_</PFX>
   <FAM Q="MD">Hochheim</FAM><DEL>-</DEL><FAM
Q="SP">Weilenfels</FAM>
   <SFX Q="PR WK">NCFSA</SFX>
</EN>
```

The next example is an organization name, "Health Level Seven, Inc." in simple string form:

```
<EN>Health Level Seven, Inc.</EN>
```

and as a fully parsed name

```
<EN>Health Level Seven<DEL>, </DEL><SFX Q="LS">Inc.</SFX></EN>
```

2.8.3 Trivial Name (TN) restricts EN

The trivial name (TN) is an entity name that consists of only one name part without any name part type or qualifier. The TN, and its single name part are therefore equivalent to a simple character string.

2.8.4 Person Name (PN) restricts EN

Since most of the functionality of entity name is in support of person names, the person name (PN) is only a very minor restriction on the entity name part qualifier.

2.8.5 Organization Name (ON) restricts EN

A name for an organization, such as "Health Level Seven, Inc." An organization name consists only of untyped name parts, prefixes, suffixes, and delimiters.

2.9 Integer Number (INT)

Integer numbers (-1,0,1,2, 100, 3398129, etc.) are precise numbers that are results of counting and enumerating. No arbitrary limit is imposed on the range of integer numbers. Two exceptional values are defined for the positive and negative infinity (see Table 2.)

2.10 Real Number (REAL)

Real numbers (, 1.5, $2.23e10^{15}$, etc.) are needed beyond integers whenever quantities of the real world are *measured*, *estimated*, or computed from other real numbers.

2.10.1 Literal Form

A real number is represented in decimal form with optional + or – sign, and optional decimal point, and optional exponential notation using a case insensitive "e" between the mantissa and the exponent. The number of significant digits must conform to the precision property.

Alternative representations two-thousand are `2000`, `2000.`, `2e3`, `2.0e+3`, `+2.0e+3`.

Note that the literal form does not carry type information. For example, "2000" is a valid representation of both a real number and an integer number. No trailing decimal point is used to disambiguate from integer numbers. An ITS that uses this literal form must recover the type information from other sources.

2.10.1.1 precision : INT

The precision property indicates the quality of the approximation of a decimal real number representation. Precision is the number of significant decimal digits in that decimal representation. The precision attribute is the precision of a decimal digit representation, *not the precision or accuracy of the real number value*. Precision does not play a role in deciding whether two real number values are equal.

The purpose of the precision property for the real number data type is to faithfully capture the whole information presented to humans in a number. The amount of decimal digits shown conveys information about the uncertainty (i.e., precision and accuracy) of a measured value.

Note: the precision of the representation is independent from uncertainty (precision accuracy) of a measurement result. If the uncertainty of a measurement result is important, one should send uncertain values as defined in Section 4.4.

The rules for what digits are significant are as follows:

All non-zero digits are significant.

All zeroes to the right of a significant digit are significant.

When all digits in the number are zero the zero-digit immediately left to the decimal point is significant (and because of rule 2, all following zeroes are thus significant too.)

Note, these rules of significance differ slightly from the more casual rules taught in school. Notably trailing zeroes before the decimal point are consistently regarded significant here. Elsewhere, e.g., 2000 is ambiguous as to whether the zeroes are significant. This deviation from the common custom is warranted for the purpose of unambiguous communication.

Examples:

2000 has 4 significant digits.

2e3 has 1 significant digit, used if one would naturally say "2000" but precision is only 1.

0.001 has 1 significant digits.
1e-3 has 1 significant digit, use this if one would naturally say "0.001" but precision is only 1.
0 has 1 significant digit.
0.0 has 2 significant digits.
000.0 has 2 significant digits.
0.00 has 3 significant digits.
4.10 has 3 significant digits.
4.09 has 3 significant digits.
4.1 has 2 significant digits.

The precision of the representation *should* match the uncertainty of the value. However, precision of the representation and uncertainty of the value are separate independent concepts. Refer to Section 4.4 for details about uncertain real numbers.

For example "0.123" has 3 significant digits *in the representation*, but the *uncertainty of the value* may be in any digit shown or not shown, i.e., the uncertainty may be 0.123±0.0005, 0.123±0.005 or 0.123±0.00005, etc. Note that external representations *should* adjust their representational precision with the uncertainty of the value. However, since the precision in the digit string is granular to ±0.5 the least significant digit, while uncertainty may be anywhere between this raster, 0.123±0.005 would also be an adequate representation for the value between 0.118 and 0.128.

ITS Note: on a character based Implementation Technology the ITS need not represent the precision as an explicit attribute if numbers are represented as decimal digit strings. In that case, the ITS must abide by the rules of an unambiguous determination of significant digits. A number representation must not produce more or less significant digits than were originally in that number. Conformance can be tested through round-trip encoding – decoding – encoding.

2.10.2 Equality

Note that a raw equality test on real numbers is unreasonable for most practical purposes, since infinitesimal equality is rarely meaningful in practice. This definition of equality is designed to be reasonably useful for simple cases. For more sophisticated cases it is recommended to compare real numbers based on intervals, that is, to test whether a real value falls within a certain range (interval).

2.11 Ratio (RTO)

A ratio is a quantity constructed through division of a numerator quantity with a denominator quantity. Ratios are *different from* "rational numbers," i.e., in ratios common factors in the numerator and denominator never cancel out. A ratio of two real or integer numbers is not automatically reduced to a real number.

Table 27: Summary of Ratio (RTO)

Name	Type	Status	Definition
numerator	QTY	mandatory	The numerator of the ratio.
denominator	QTY	mandatory	The denominator of the ratio

The QTY data type is an abstract generalization that stands for INT, REAL, PQ, and MO.

The purpose of the ratio data type is to support certain quantities produced by laboratories, such as *titers* (e.g., "1:128"). Ratios are not simply "structured numerics;" blood pressure measurements (e.g. "120/60") are not ratios.

Note: This data type is not defined to generally represent rational numbers. In this Ratio data type, it is *not* correct to cancel out common factors in numerator and denominator. For example, if a ratio is recorded as 2:8, it should not be reduced to 1:4.

The default value for both numerator and denominator is the integer number 1 (one.) The denominator may not be zero.

2.11.1 Literal Form

A ratio literal consists of a numerator, a colon (as separator), followed by the denominator. When the colon and denominator are missing, the integer number 1 is assumed as the denominator.

2.12 Physical Quantity (PQ)

A physical quantity is a dimensioned quantity expressing the result of a measurement act. It consists of

Table 28: Summary of Physical Quantity (PQ)

Name	Type	Status	Definition
value	REAL	mandatory	The magnitude of the quantity measured in terms of the unit
unit	CS	mandatory	The unit of measure

The codes for unit of measure are specified in the *Unified Code for Units of Measure* (UCUM) [http://aurora.rg.iupui.edu/UCUM]. The default unit of measure is 1 (the "unity").

Note that equality of physical quantity does not require the values and units to be equal independently. Value and unit is only how we represent physical quantities. For example, 1 m equals 100 cm. Although the units are different and the values are different, the physical quantities are equal! Therefore one should never expect a particular unit for a physical quantity but instead provide automated conversion between different comparable units. (See Part II for more information.)

2.12.1 Literal Form

Physical quantities are expressed as a real number followed by optional white space and a character string representing a valid code in the *Unified Code for Units of Measure*. For example "`12 m`" for 12 meter.

2.13 Monetary Amount (MO)

A monetary amount is a quantity expressing the amount of money in some currency. Currencies are the units in which monetary amounts are denominated in different economic regions.

Table 29: Summary of Monetary Amount (MO)

Name	Type	Status	Default	Constraint	Definition
value	REAL	mandatory	NULL		The magnitude of the monetary amount in terms of the currency unit.
currency	CS	mandatory	NULL	ISO 4217	The currency unit

Note: monetary amounts are usually precise to 0.01 (one cent, penny, paisa, etc.) For large amounts, it is important not to store monetary amounts in floating point registers, since this may lose precision. However, this specification does not define the internal storage of real numbers as fixed or floating point numbers.

The precision attribute of the real number type is the precision of the decimal representation, not the precision of the value. The real number type has no notion of uncertainty or accuracy. For example, "1.99 USD" (precision 3) times 7 is "13.93 USD" (precision 4) and should not be rounded to "13.9" to keep the precision constant.

Table 30: Selected ISO 4217 currency codes

Country	Currency	Code
Argentina	Argentine Peso	ARS
Australia	Australian Dollar	AUD
Brazil	Brazilian Real	BRL
Canada	Candian Dollar	CAD
Chile	Unidades de Formento	CLF
China	Yuan Renminbi	CNY
European Union	Euro	EUR
European Union	ECU (until 1998-12-31)	XEU
Finland	Markka	FIM
France	French Franc	FRF
Germany	Deutsche Mark	DEM
India	Indian Rupee	INR
Israel	Shekel	ILS
Japan	Yen	JPY
Mexico	Mexican Nuevo Peso	MXN
Netherlands	Netherlands Guilder	NLG
New Zealand	New Zealand Dollar	NZD
Philippines	Philippine Peso	PHP
Russian Federation	Russian Ruble	RUR
South Africa	Rand	ZAR
Spain	Spanish Peseta	ESP
Switzerland	Swiss Franc	CHF
Thailand	Baht	THB
United Kingdom	Pound Sterling	GBP
United States	US Dollar	USD

2.13.1 Literal Form

Monetary amounts are represented by an amount, optional white space, and the currency code. For example, "`189.95 USD`" for 189.95 U.S. Dollar.

2.14 Point In Time (TS)

A time-stamp represents a point in time. As nobody knows when time began, a point in time is expressed as the amount of time that has elapsed from some arbitrary zero-point, called an epoch. So when we assign the time stamp "198910291030" to an event, we are saying that 1989 years, 10 months, 29 days, 10 hours and 30 minutes have elapsed since the epoch (the birth of Christ).

2.14.1 Literal Form

Time stamps begin with the 4-digit year (beginning counting at zero); followed by the 2-digit month of the year (beginning counting at one); followed by the 2-digit day of the month (beginning with one); followed by the 2-digit hour of the day (beginning with zero); and so forth. For example, "`200004010315`" is a valid expression for April 1, 2000, 3:15 am.

A calendar expression can be of variable precision, omitting parts from the right.

For example, "`20000401`" is precise only to the day of the month.

The last calendar unit may be written as a real number, with the number of integer digits specified, followed by the decimal point and any number of fractional digits.

For example, "`20000401031520.34`" means April 1, 2000, 3:15 and 20.34 seconds.

The optional time zone suffix begins with a plus (+) or minus (–) followed by digits for the hour and minute cycles. UTC is designated as offset "+00" or "-00" (the ISO 8601 and ISO 8824 suffix "Z" for UTC is not permitted.)

For example, 20000401031520.34-0500 indicates April 1, 2000, 3:15 and 20.34 Eastern Standard Time.

3 Generic Collections

This section defines data types that can "collect" other data values, Set, Sequence, Bag and Interval.

3.1 Set (SET)

A set is a value that contains other values of a certain data type as its elements. The elements are contained in no particular ordering. All elements in the set are distinct, the same element value can not be contained more than once in the set. A set may only contain non-NULL elements. Exceptional values (NULL-values) can not be elements of a set.

3.2 Sequence (LIST)

A sequence is an ordered collection of discrete values.

3.3 Bag (BAG)

A bag is an unordered collection of elements where each element can be contained more than once in the bag.

3.4 Interval (IVL)

An interval is a set of consecutive values of any ordered data type. An interval is thus a contiguous subset of its base data type.

Some examples:

- There is an interval between 2 and 4 meters.
- There is an interval between December 4, 2000, 10:00 am and 10:30 am
- There is an interval between mile markers 210 and 225 on the Pennsylvania Turnpike.

Note that the interval boundaries must be of comparable types. It is nonsensical to describe the interval between 2 meters and 4 seconds.

Table 31: Summary of Interval (IVL)

Name	Description
low	This is the low boundary of the interval.
high	This is the upper boundary of the interval.
width	The width is the difference between high and low boundary.
center	The center is defined of finite intervals and is then the arithmetic mean of the interval (low plus high divided by 2).
lowClosed	A Boolean, indicating whether the interval is closed or open at the low boundary. For a boundary to be closed, a finite boundary must be provided, i.e. unspecified or infinite boundaries are always open.
highClosed	A Boolean, indicating whether the interval is closed or open at the high boundary.

In any interval representation only two of the four properties high, low, width and center need to be stated and the other two can be derived. Incomplete intervals exist, where only one property is values, particularly, when no boundary or center is known, the width may still be known. For example, one knows that an activity takes about 30 minutes, but one may not yet know when that activity is started.

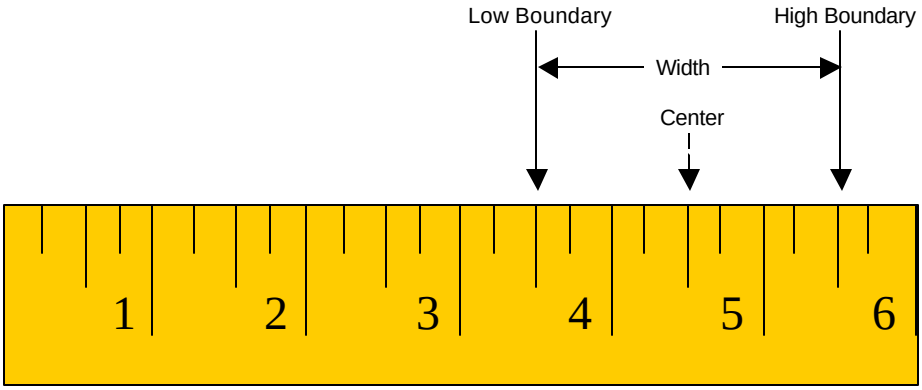

Figure 1: Properties of the interval 3.5–5.5

3.4.1 Literal Form

The literal for the interval data type is defined such that it is as intuitive to humans a possible. Five different forms are defined:

1) The interval form using square brackets, e.g., "`[3.5; 5.5[`" meaning $3.5 \le n < 5.5$. Where the two points of the bracket point to the boundary value, the boundary is closed, meaning the boundary value is finite and included in the interval. 3.5, the low boundary in this example, is closed. Where the two points of the bracket point away from the boundary value, the boundary is open, meaning that either it is infinite or not included in the interval. 5.5, the high boundary in this example, is open, meaning that values up to, but not including, 5.5 are included in the interval. A semicolon separates boundaries.

2) The dash-form, e.g., "`3.5-5.5`" meaning $3.5 \le n \le 5.5$. .All values, between and including 3.5 and 5.5 are included. In other words, both boundaries are closed. "3.5–5.5" is equivalent to "`[3.5; 5.5]`".

3) The "comparator" form, using relational operator symbols, e.g., "`<5.5`".

4) The center-width form, e.g., "`4.5[2.0[`". 4.5 is the center point of the interval, 2.0 is the width, the low boundary is closed and the upper boundary is open.

5) The width-only form using square brackets, e.g., "`[2.0[`", meaning $m\text{-}1.0 \le n < m\text{+}1.0$

The reason for defining multiple representations is that each has its pros and cons. Part II of this document explains what they are.

Table 32: Examples of interval literals

literal	low		high		alternate	
	closed	low	high	closed	center	width
`3.5-5.5`	true	3.5	5.5	true	4.5	2.0
`[3.5;5.5]`	true	3.5	5.5	true	4.5	2.0
`[3.5;5.5[`	true	3.5	5.5	false	4.5	2.0
`4.5[2.0]`	true	3.5	5.5	true	4.5	2.0
`4.5[2.0[`	true	3.5	5.5	false	4.5	2.0
`<5.5`	false	–∞	5.5	false	N/A	∞
`>3.5`	false	3.5	∞	false	N/A	∞
`>=3.5`	true	3.5	∞	false	N/A	∞
`<=5.5`	false	–∞	5.5	true	N/A	∞
`]-inf;5.5]`	false	–∞	5.5	true	N/A	∞
`[3.5;+inf[`	true	3.5	∞	false	N/A	∞
`];5.5]`	false	UNK	5.5	true	UNK	UNK

[3.5;[	true	3.5	UNK	false	UNK	UNK
-3.5-3.5	true	–3.5	3.5	true	0.0	7.0
-5.5--3.5	true	–5.5	–3.5	true	–4.5	2.0
[-5.5;-3.5]	true	–5.5	–3.5	true	–4.5	2.0
-4.5[2.0]	true	–5.5	–3.5	true	–4.5	2.0
<-3.5	false	–∞	–3.5	false	N/A	∞
>-5.5	false	–5.5	∞	false	N/A	∞
[3.5;3.5]	true	3.5	3.5	true	3.5	0
[2.5]	true	UNK	UNK	true	UNK	2.5
[2.5[	true	UNK	UNK	false	UNK	2.5

3.4.1.1 Interval of Physical Quantities (IVL<PQ>)

An interval of physical quantities is represented by augmenting the interval representation with unit of measurement code(s). If the boundaries are expressed in the same unit of measure (typical), then the unit of measure may be appended to the end of the interval, with a single white space separator in between. (e.g., "3.5-5.5 cm").

Alternatively, a unit of measure may be appended to each boundary (again, separated by a white space). The interval "35 mm-5.5 cm" is the same as "3.5-5.5 cm".

For another example: "`[0;5] mmol/L`" or "`<20 mg/dL`" are valid literal forms of intervals of physical quantities. The generic interval form, e.g., "`[50 nm; 2 m]`" is also allowed.

3.4.1.2 Interval of Points in Time (IVL<TS>)

The literal form for interval of point in time is exceptional.

- The "dash form" is not allowed for intervals of point in time (to avoid conflicts with the timezone notation)
- A "hull form" is defined instead. For example, "`19870901..19870930`" is a valid literal using the hull-form. The value is equivalent to the interval form "`[19870901;19871001[`". (An "interval hull" is an interval that snugly encloses two intervals.)

The hull-form further allows an abbreviation. For example "`19870512..23`" means May 12, 1987 to May, 23, 1987. Notice how the timestamp on the right-hand side does not need to repeat digits on the left-hand side that are the same as for the right-hand side timestamp. The two timestamps are right-aligned and the digits to the left copied from the lower to the higher timestamp. This is a simple string operation. However, note that May 12, 1987 to June 2, 1987 is "`19870512..0602`", and not "`20000512..02`".

4 Generic Type Extensions

Generic type extensions enable comments of various natures to be attached to any data value of any data type. These generic type extensions inherit most properties of their base type and add some specific feature to it.

There are several generic data type extensions:

- History (HIST) and History Item (HXIT)
- Uncertain Value – Probabilistic (UVP) and Non-Parametric Probability Distribution (NPPD)
- Parametric Probability Distribution (PPD)

We'll examine each in varying levels of detail.

At this time HL7 does not permit use of generic type extensions, except where explicitly enabled (in this or another HL7 specification) for such use cases where this advanced functionality is important.

4.1 History (HIST) and History Item (HXIT)

4.1.1 History Item (HXIT)

A generic data type extension that allows one to tag a time range to any data value of any data type. The time range is the time in which that data is (was) valid.

Table 33: Summary of History Item (HXIT)

Name	Type	Status	Definition
…	…	…	*The data of the extended value.*
validTime	IVL<TS>	optional	The time interval during which the given information was, is, or is expected to be valid.

For example, a value of type HXIT<CS> extended from a Coded Simple Value (CS) data type has all of the properties of the original CS value, plus a time range indicating when that code is or was valid.

Data types that already have a valid time range property (i.e., II, AD, TEL) obviously do not need these extensions. Their valid time range property can be mapped to the valid time property of the HXIT, in fact, those data types are considered history items by themselves. For example, II is the same data type as HXIT<II>.

4.1.2 History (HIST)

A generic data type that collects an entire history of data values. A history is a set history items (HXIT)

For example, consider an employee number of type Instance Identifier (II). During the employee's career, his institution may be acquired, sold, reacquired, merged, etc. Each such transaction may result in a new employee number for the same person, and it might be necessary to produce a historical list of all such numbers. The HIST extension to employee number enables this use case to be realized.

The history information is not limited to the past; expected future values can also appear.

4.2 Uncertain Value – Probabilistic (UVP)

A generic data type extension used to specify a probability expressing the information producer's belief that the given value holds.

Table 34: Summary of Uncertain Value – Probabilistic (UVP)

Name	Type	Status	Definition
…	…	…	*The data of the extended value.*
probability	REAL	optional	The probability assigned to the value, a real number between 0 (certainly not) and 1 (certain).

Probabilities are subjective and (as any data value) must be interpreted in their individual context, for example, when new information is found the probability might change. Thus, for any message (document, or other information representation) the information – and particularly the probabilities – reflect what the information producer believed was appropriate for the purpose and at the time the message (document) was created.

For example, at the beginning of the 2000 baseball season (May), the Las Vegas odds makers may have given the New York Yankees a probability of 1 in 10 (0.100) of winning the World Series. At the time of this writing, the Yankees and Mets have won their respective pennants, but the World Series has yet

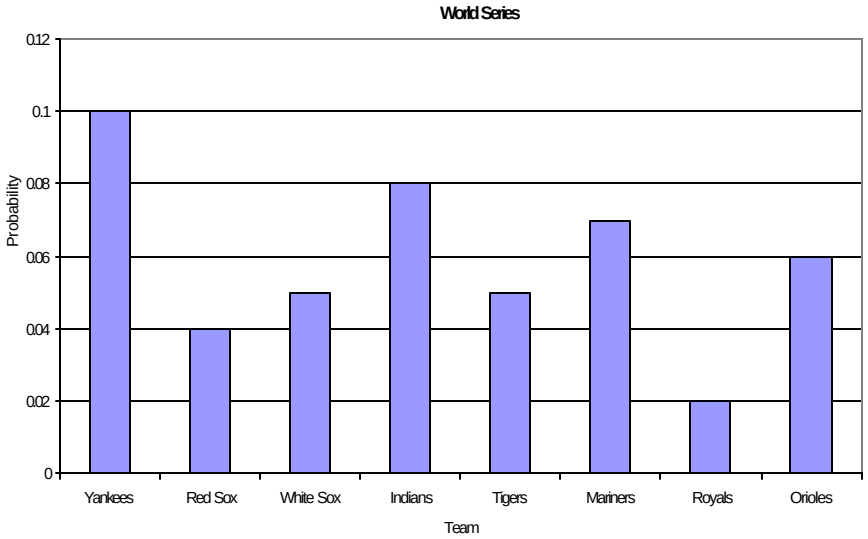

Figure 2: Example of a Histogram

to begin. The probability of the Yankees winning the World Series is obviously significantly greater at this point in time, perhaps 6 in 10 (0.600). The context, and in particular the time of year, made all the difference in the world.

Since probabilities are subjective measures of belief, they can be stated without being "correct" or "incorrect" *per se*, let alone "precise" or "imprecise". Notably, one does not have to conduct experiments to measure a frequency of some outcome in order to specify a probability. In fact, whenever statements about individual people or events are made, it is not possible to confirm such probabilities with "frequentists" experiments.

Returning to our example, the Las Vegas odds makers can not insist on the Yankees and Mets playing 1000 trial games prior to the Series; even if they could, they would not have the fervor of the real Series and therefore not be accurate. Instead, the odds makers must derive the probability from past history, player statistics, injuries, etc.

4.3 Non-Parametric Probability Distribution (NPPD)

This is a generic data type that collects multiple uncertain values with probabilities as a histogram. The easiest way to visualize this is a bar chart as shown in Figure 2.

This example illustrates the probability of selected major league baseball teams winning the World Series (prior to the season start). Each team is mutually exclusive, and were we to include all of the teams, the sum of the probabilities would equal 1 (i.e., it is certain that one of the teams will win).

Semantically, a non-parametric probability distribution contains *all* possible values and assigns probabilities to each of them. Our example has left out quite a few teams. The rules for this data type tell us that the other teams would share the "leftover" probability equally. The 8 teams in the example have a collective probability of winning the World Series of 0.47. If there were a total of 24 teams in the league, then 16 are not shown. Each of these teams would be assigned a probability of (1.00 – 0.47) ÷ 16 = 0.033125.

4.4 Parametric Probability Distribution (PPD)

A generic data type extension that assigns probability along a continuous scale according to parameters.

The PPD is typically visualized as a continuous line or area graph. The X-axis of the PPD represents a continuous scale, as opposed the mutually exclusive discrete alternatives of the NPPD. The most

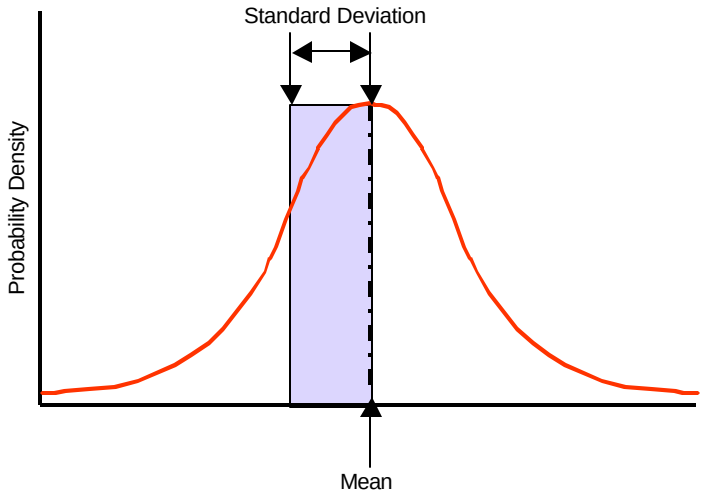

Figure 3: Example of a Continuous Probability Distribution

obvious example is the normal distribution ("bell curve" see Figure 3.) By specifying the parameters mean and standard deviation (which specifies how "skinny or fat" the bell is), one can convey a probability or statistical distribution.

For example, the most common college entrance exam in the United States is the SAT, which is comprised of two parts: verbal and math. Each part has a minimum score of 400 (no questions answered correctly) and a perfect score of 800. In 1998, according to the College Board, 1,172,779 college-bound seniors took the test. The mean score for the math portion of the test was 512, and the standard deviation 112. These parameter values (512, 112), tagged as the normal distribution parameters, paint a pretty good picture of test score distribution. In most cases, there is no need to specify all 1-million+ points of data when just 2 parameters will do!

Note that the normal distribution is only one of several distributions defined for HL7. A list of distributions and their associated parameters, as well as conformance criteria appears in Part II of this specification.

4.4.1 Literal

PPD are represented as the mean value, white space, left parenthesis, distribution code, standard deviation, and right parenthesis. Our SAT example distributions would be represented as "512 (N112)"

This subject is treated in much greater detail in Part II of this specification.

5 Timing Specification

The timing specification suite of data types is used to specify the complex timing of events and actions such as those that occur in order management and scheduling systems. It also supports the cyclical validity patterns that may exist for certain kinds of information, such as phone numbers (evening, daytime), addresses (so called "snowbirds," residing in the south during winter and north during summer) and office hours.

The timing specification data types include point in time (TS) and the interval of time (IVL<TS>) (defined in Sections 2.14 and 3.4 above) and add types that are specifically suited to repeated schedules. These additional types include periodic interval, event-related periodic interval, and finally the generic timing specification types itself. All these timing types describe the time distribution of repeating states or events.

5.1 Periodic Interval of Time (PIVL)

The periodic interval of time specifies an interval of time that recurs periodically. Periodic intervals have two properties, phase and period. The phase specifies the "interval prototype" that is repeated every period.

The phase also marks the anchor point in time for the entire series of periodically recurring intervals. The recurrence of a periodic interval has no beginning or ending (infinite in both future and past)

Table 35: Summary of Periodic Interval of Time (PIVL<TS>)

Name	Type	Status	Definition
phase	IVL<TS>	mandatory	A prototype of the repeating interval, may anchor the periodic interval sequence at a certain point in time.
period	PQ (~1 s)	mandatory	A time duration specifying the frequency at which the periodic interval repeats.
alignment	CS	optional	Specifies an alignment of the repetition to a calendar (e.g., to distinguish every 30 days from "the 5th of every month".)
institutionSpecifiedTime	BL	optional	Indicates whether the exact timing is up to the party executing the schedule (e.g., to distinguish "every 8 hours" from "3 times a day".)

This is simpler to explain with examples:

Every other Thursday from 7am to 8am, starting with November 30, 2000. The phase is November 30, 2000 7 to 8 am ("`[20001130070;20001130080[`"). The period is two weeks. This phase anchors the bi-weekly repetition cycle at November 30, followed by December 14, 28, etc. Note that the phase-interval repeats into the future and into the past. The November 30, 2000, is only the anchor-point of the repetition; it is not necessarily where the repetition starts. In this example, the repetition could have started November 16 2000, or even earlier than that.

This example of a periodic interval is fully specified because both the period and the phase with an anchor are fully specified. The phase interval may be only partially specified, without an anchor, and where either only the width or only one boundary may be specified.

Every two weeks for one hour. The phase is the interval one hour; the period is two weeks. Only the width of the phase interval is known, so there is no anchor. This means, the exact day and time every two weeks is not specified.

Every eight hours starting (for example) on November 30, 2000 at 4 o'clock. The phase is an interval starting any day at 4 o'clock. The period is eight hours. Only the phase's low boundary but not the phase's high boundary is specified.

Every eight hours for ten minutes starting at 4 o'clock. This is fully specified since the period, and both the phase's low boundary and width are specified (low boundary and width implies the high boundary, 4:02.)

Remember that the phase interval is just a prototype of the repeating period, even though the phase interval may be an exact date and time. So, in the last example, phase interval could just as well begin at 8 am and ends at 8:10 am or begin at midnight and end at 12:10 am. And that phase interval could be at any day in the past or future. The phase interval is only a prototype of the periodically repeating interval.

Oftentimes repeating schedules are only approximately specified. For instance "three times a day for ten minutes each" does not usually mean a period of precisely 8 hours and does often not mean exactly 10 minutes intervals. Rather the distance between each occurrence may vary as much as between 3 and 12 hours and the width of the interval may be less than 5 minutes or more than 15 minutes. An uncertain periodic interval can be used to indicate how much leeway is allowed or how "timing-critical" the specification is.

5.1.1 Literal form

There are several forms of representation for periodic intervals of time:

- generic form, continuous: ⟨*phase* : IVL⟨T⟩⟩ **/** ⟨*period* : T.diff⟩
- generic form, calendar aligned: ⟨*phase* : IVL⟨T⟩⟩ **/** ⟨*period* : T.diff⟩ **@** ⟨*alignment*⟩ [**IST**].
- calendar pattern, calendar aligned: ⟨*anchor*⟩ [**..** ⟨*calendar digits*⟩]] **/** ⟨*number* : INT⟩ [**IST**]

5.1.1.1 Generic Form, Continuous

The generic form, continuous, is:

⟨*phase* : IVL⟨T⟩⟩ **/** ⟨*period* : T.diff⟩ [**IST**]

By "continuous," we mean that the period is exactly the same throughout the duration of the periodic interval of time.

Let's dissect an example:

```
[200004181100;200004181110]/(7 d)
```

specifies every Tuesday from 11:00 to 11:10 AM.

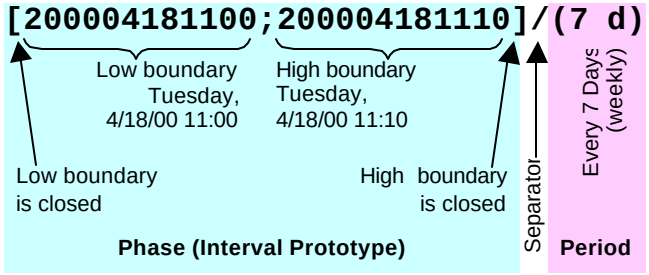

Figure 4: Dissection of a Periodic Interval

The Phase is an IVL<TS> value conveying a range of time. In this example, the 10-minute interval ranging from 11:00 to 11:10 on 4/18/00 is specified. Refer back to Section 3.4 on page 24 for details on how a range is expressed.

The Period is expressed as a quantity of time. Refer back to Section 2.12 for details on expressing quantities. The unit of time code (in this case "d" for day) is found in The Unified Code for Units of Measure at http://aurora.rg.iupui.edu/UCUM.

Note that this is a continuous periodic interval of time because the period must always be 7 days. This periodic interval specifies every Tuesday from 11:00 to 11:10 am, because the 18th day of April 2000 was a Tuesday and Tuesdays follow each other every 7 days.

5.1.1.2 Generic form, calendar aligned

The generic calendar aligned form is used when the period may vary over the duration due to the nature of the calendar (e.g., some months have 31 days, some 30, some 28, etc.).

⟨*phase* : IVL<T>⟩ **/** ⟨*period* : T.diff⟩ **@** ⟨*alignment*⟩] [**IST**].

Let's dissect another example:

```
[200004181100;200004181110]/(1 mo)@DM
```

specifies every 18th of the month 11:00 to 11:10 AM.

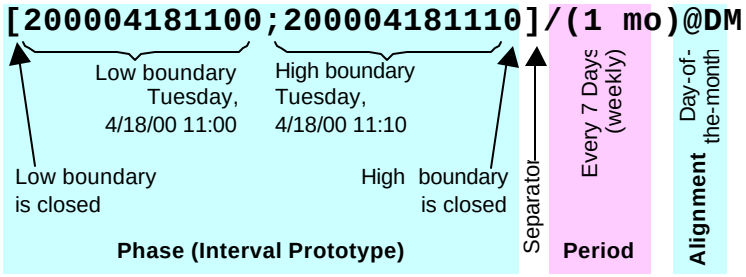

Figure 5: Dissection of a periodic interval with calendar alignment

The phase is the same interval as in the first example. The period is one month (about 29 days average), but the calendar alignment is set to "day of the month" (DM). This means, the interval will be repeated at the same day of the month, in this example, the 18th of every month. This is *not* a continuous interval because the period varies; the first period is 30 days, the second 31 days, and the third 30 days, and so on. The alignment code is normally one of the codes shown in Table 36:

Table 36: Calendar Periods for Alignment

Name	Code
month of the year	MY
week of the year	WY
day of the month	DM
day of the year	DY
day of the week (begins with Monday)	DW
hour of the day	HD
minute of the hour	NH
second of the minute	SN

5.1.1.3 Calendar Pattern Form

The calendar pattern form is used to specify calendar-aligned timing more intuitively using calendar patterns. A calendar pattern is a calendar date where the higher significant digits (e.g., year and month) are omitted. In order to interpret the digits, a period identifier is prefixed that identifies the calendar period of the left-most digits. This calendar period identifier anchors all the calendar digits following to the right.

Table 37: Calendar Periods for Calendar Pattern

Name	Code
year	Y
month of the year	M
day of the month	D
hour of the day	H
minute of the hour	N
second of the minute	S

For example: "M0219" is February 19 the entire day every year. This periodic interval has the February 19 of any year as its phase, a period of one year, and alignment month of the year (MY). "M02191100..1110" is February 19, 11:00 to 11:10 AM, every year.

As with the TS literal, intervals can be specified using the two periods ".." followed by calendar digits matching the preceding calendar expression from the right (see interval of point in time, Section 3.4.1.2)

As with the TS, the calendar pattern may omit digits on the right. When digits are omitted on the right, this means the interval from lowest to highest for these digits.

Table 38: Examples for literal expressions for periodic intervals of time

Generic Form	Calendar Pattern Form	Description
[198709;198710[/(1 a)@MY	M09	September, the entire month, every year (note that in the year 1987 in the generic form is irrelevant since the periodic interval recurs every year past and future.)
[19870915;19870916[/(1 a)@DM	M0915	September 15, the entire day, every year
[1987091516;1987091517[/(1 a)@DM	M091516	September 15 at 4 PM, the entire hour, every year
[198709151630;198709151710]/(1 a)@DM	M09151630..1710	September 15 at 4:30 5:10 PM, every year
[1987091516;[/(1 a)@DM		September 15 at 4 PM, end time explicitly unknown, every year
[198709151630;198709151631[/(1 a)@DM	M09151630	September 15 at 4:30 PM, the entire minute, every year
[1987091516;1987091517[/(1 mo)@DM	D1516..17	every 15th day of the month at 4 to 5 PM
[1987091516;1987091517[/(1 mo)		September 15, 1987 from 4 to 5 PM and then every 730.5 hours continuously (this example has little practical value beyond comparing the unaligned with the aligned form in the preceding row.)
[1987091516;1987091517[/(1 mo)@HD		September 15, 1987 from 4 to 5 PM and then every 30.4375 days, but aligned to the hour of the day.
[1 mo]/(2 mo)@MY	M/2	every other month of the year; (Jan, Mar, …) vs. (Feb, Apr, …) is undefined
[198701;197502[/(2 mo)@MY	M01..12/2	every other month of the year, Jan, Mar, …
[198702;197503[/(2 mo)@MY	M02..12/2	every other month of the year, Feb, Apr, …
[19870401;19870930[/(1 a)@DM	M04..09	April 1until (and including) September 30
19870401-0930/(1 a)@DM	M0401..0930	April 1 to September 30 (the generic form uses the dash-form for the phase interval)
[20001202;20001203[/(1 wk)@DW	J6	every Saturday
[20001202;20001203[/(2 wk)@DW	J6/2	every other Saturday
[20001202;20001203[/(3 wk)@DW	J6/3	every third Saturday
[1 d]/(2 d)@DW	J/2	every other day of the week; (Mon, Wed, Fri, …) vs. (Tue, Thu, Sat, …) is undefined
[20001204;20001205[/(2 d)@DW	J2..6/2	every other day of the week (Tue, Thu, Sat, Tue, Thu, Sat, …)
[20001204;20001205[/(2 d)	D/2	every other day (Tue, Thu, Sat, Mon, Wed, Fri, Sun, Tue, …)
[19870601;19870606[/(1 wk)@DW	J1..5	Monday to Friday every week
[19870601;19870608[/(2 wk)	W/2	every other week (continuous)
[19870101;19870105[/(2 wk)@WY	WY/2	every other week of the year (a blunt example on the impact of the calendar alignment: the phase interval spans only 4 days and yet it represents an entire week in the calendar alignment “week of the year”.)
[19870406;19870413[/(1 a)@WY	WY15	the 15th calendar week of every year
[19870105;19870112[/(1 mo)@WM	WM2	the second week of the month, every month
[19870508;19870509[/(1 a)@DY	DY128	the 128th day of the year, every year
[10 min]/(2 d)		every other day for 10 minutes (only width of repeating interval is known)
[1 h]/(8 h)	H/8	every eighth hour (each time a 60 minutes interval)
[1 h]/(8 h) IST	H/8 IST	three times a day at institution specified times (each time a 60 minutes interval)
/(8 h) IST		three times a day at institution specified times. Nothing about the repeating interval is known i.e., this includes only a period (frequency), while the phase is left undefined

A calendar pattern followed by a slash and an integer number *n* indicates that the given calendar pattern is to apply every n^{th} time.

For example: "`D19/2`" is the 19^{th} of every second month.

A calendar pattern expression is evaluated at the time the pattern is first enacted. At this time, the calendar digits missing from the left are completed using the earliest date matching the pattern (and following a preceding pattern in a combination of time sets).

For example: "`D19/2`" is the 19^{th} of every second month. If this expression is evaluated on March 14, 2000 the phase is completed to: "`[20000319;20000320[/(2 mo)@DM`" and thus the two-months cycle begins with March 19, followed by May 19, etc. If the expression were evaluated by March 20, the cycle would begin at April 19, followed by June 19, etc.

If after the calendar period identifier no calendar digits follow, the pattern matches any date. The integer number following the slash indicates the length of the cycle.

For example: "`CD/2`" is every other day, "`H/8`" is every 8^{th} hour.

A calendar pattern may be followed by the three letters "`IST`" to indicate that within the larger calendar cycle (e.g., day for hour of the day) the repeating events are to be appointed at *institution specified times*. This is used to specify such schedules as "three times a day" where the periods between two subsequent events may vary well between 4 hours (between breakfast and lunch) and 10 hours (over night.)

5.2 Event-Related Periodic Interval of Time (EIVL)

The event-related periodic interval of time allows specifying a periodic interval of time based on activities of daily living, important events that are time-related but not fully determined by time.

Table 39: Summary of Event Related Interval of Time (EIVL<TS>)

Name	Type	Status	Definition
event	CV	mandatory	A code for a common (periodical) activity of daily living
offset	IVL<PQ>	mandatory	An interval that marks the offsets for the beginning, width and end of the event-related periodic interval measured from the time each such event actually occurred

For example, "one hour after breakfast" specifies the beginning of the interval at one hour after breakfast is finished. Breakfast is assumed to occur before lunch but is not determined to occur at any specific time.

Table 40: Event Codes for Event-Related Periods

Code	Definition
HS	the hour of sleep (e.g., H18-22)
AC	before meal (from lat. *ante cibus*)
PC	after meal (from lat. *post cibus*)
IC	between meals (from lat. *inter cibus*)
ACM	before breakfast (from lat. *ante cibus matutinus*)
ACD	before lunch (from lat. *ante cibus diurnus*)
ACV	before dinner (from lat. *ante cibus vespertinus*)
PCM	after breakfast (from lat. *post cibus matutinus*)
PCD	after lunch (from lat. *post cibus diurnus*)
PCV	after dinner (from lat. *post cibus vespertinus*)
ICM	between breakfast and lunch
ICD	between lunch and dinner
ICV	between dinner and the hour of sleep

For example, one hour after meal would be "`PC+[1h;1h]`". One hour before bedtime for 10 minutes: "`HS-[50min;1h]`".

5.3 General Timing Specification (GTS)

The previous sections discussed simple and periodic intervals of time that can express simple repeating schedules. Real-world applications often require more specificity and more complex timing. Recall that a periodic interval of time implies no beginning and no end. Obviously, orders, scheduling, and other time-dependent applications require a beginning and end. Likewise, there may be the need to add a time period not included in the interval (a "one-time occurrence"), or to exclude a specific time period (such as a holiday).

The GTS data type accommodates these needs by enabling one to combine multiple intervals of time, using intersection, union, and exclusion operations. The following operators are used:

Table 41: GTS Set-Operators

Operation	Operator	Priority
Intersection	" " (white space)	high
Union	";" (semicolon)	low
Exclusion	"\" (back slash)	low
Periodic Hull	".." (two periods)	high

Also parentheses can be used to overcome operator precedence when necessary.

So, let's take an example: "Monday to Thursday 8 AM to 4 PM and Friday 8 AM to 12 noon, from October 1, 2000 to October 23, 2000."

Logically, we are defining

- the intersection of the time interval "October 1, 2000 to October 23, 2000" (pale yellow, in the diagram),
- with the union of two calendar-aligned period intervals:
- Weekly, every Monday through Thursday 8 – 4 (blue, in the diagram), and
- Weekly, every Friday 8-12 (red, in the diagram).

The time specified by this expression is depicted in Figure 6 as all points in the checkerboard pattern.

The expression for this example (in calendar pattern form) is:

```
"20001001..23 (J1..4 H0800..1600; J5 H0800..1200)"
```

The following table contains additional examples for complex GTS literals

Table 42: Examples for Literal Expressions for Generic Timing Specifications

Literal Expression	Meaning
`M09 D15 H16 N30 S34.12`	September 15 at 4:30:34.12 PM as the intersection of multiple periodic intervals of times (calendar patterns)
`M0915163034.12`	September 15 at 4:30:34.12 PM as one simple periodic interval of time (calendar pattern)
`M01; M03; M07`	January, March, and July (a union of three periodic intervals of time)
`M04..09 M/2`	Every second month from April to September (April, June, August)
`J1; J2; J4`	Monday, Tuesday, Thursday
`W/2 J2`	every other Tuesday (intersection of every other week and every Tuesday)
`1999 WY15`	the 15th calendar week in 1999 (period code is optional for the highest calendar unit)
`WM2 J6`	Saturday of the 2nd week of the month
`M05 WM2 J6`	Saturday of the 2nd week of May
`M05 DM08..14 J7`	Mother's day (second Sunday in May.)
`J1..5 H0800..1600`	Monday to Friday from 8 AM to 4 PM
`J1..4 H0800..1600; J5 H0800..1200`	Monday to Thursday 8 AM to 4 PM and Friday 8 AM to 12 noon.

`[10 d] H/8`	Three times a day over 10 days (each time a 60 minutes interval).
`H0800..1600 \J3`	Every day from 8 AM to 4 PM, except Wednesday.
`(M0825..31 J1)..M0831`	The last calendar week of August.

The following Table 43 defines symbolic abbreviations for GTS values that can be used in GTS literals instead of their equivalent GTS term. Abbreviations are defined for common periods of the day (AM, PM), for periods of the week (business day, weekend), and for holidays. The computation for the dates of some holidays, namely the Easter holiday, involve some sophistication that goes beyond what one would represent in a GTS literal term. It is assumed that the dates of these holidays are drawn from some table or some generator module that is outside the scope of this specification.

Table 43: Abbreviations for General Timing Specifications

Code	Definition	Equivalent
AM	Every morning at institution specified times.	`H00..11 IST`
PM	Every afternoon at institution specified times.	`H12..23 IST`
`BID`	two times a day at institution specified time	`H/12 IST`
`TID`	three times a day at institution specified time	`H/8 IST`
`QID`	four times a day at institution specified time	`H/6 IST`
JB	Regular business days (Monday to Friday excluding holidays)	`J1..5 \JH`
JE	Regular weekends (Saturday and Sunday excluding holidays)	`J6..7 \JH`
JH	Holidays	
	Christian Holidays (Roman/Gregorian "Western" Tradition.)	
JHCHRXME	Christmas Eve (December 24)	`M1224`
JHCHRXMS	Christmas Day (December 25)	`M1225`
JHCHRNEW	New Year's Day (January 1)	`M0101`
JHCHREAS	Easter Sunday. The Easter date is a rather complex calculation based on Astronomical tables describing full moon dates. Details can be found at [http://www.assa.org.au/edm.html, and http://aa.usno.navy.mil/AA/faq/docs/easter.html]. Note that some Eastern Orthodox Holidays are based on the Julian calendar.	
JHCHRGFR	Good Friday, is the Friday right before Easter Sunday.	
JHCHRPEN	Pentecost Sunday, is seven weeks after Easter (the 50th day of Easter.)	
JHNUS	**United States National Holidays** (public holidays for federal employees established by U.S. Federal law 5 U.S.C. 6103.)	
JHNUSMLK	Dr. Martin Luther King, Jr. Day, the third Monday in January.	`M0115..21 J1`
JHNUSPRE	Washington's Birthday (Presidential Day) the third Monday in February.	`M0215..21 J1`
JHNUSMEM	Memorial Day, the last Monday in May.	`M0525..31 J1`
JHNUSMEM5	Friday before Memorial Day Weekend	`M0522..28 J5`
JHNUSMEM6	Saturday of Memorial Day Weekend	`M0523..29 J6`
JHNUSMEM7	Sunday of Memorial Day Weekend	`M0524..30 J7`
JHNUSIND	Independence Day (4th of July)	`M0704`
JHNUSIND5	Alternative Friday before 4th of July Weekend [5 U.S.C. 6103(b)].	`M0703 J5`
JHNUSIND1	Alternative Monday after 4th of July Weekend [5 U.S.C. 6103(b)].	`M0705 J1`
JHNUSLBR	Labor Day, the first Monday in September.	`M0901..07 J1`
JHNUSCLM	Columbus Day, the second Monday in October.	`M1008..14 J1`
JHNUSVET	Veteran's Day, November 11.	`M1111`
JHNUSTKS	Thanksgiving Day, the fourth Thursday in November.	`M1122..28 J4`
JHNUSTKS5	Friday after Thanksgiving.	`M1123..29 J5`

Note: holidays are locale-specific. Exactly which religious holidays are subsumed under JH depends on the local rules. For global interoperability, using constructed GTS expressions is safer than named holidays. However, some holidays that depend on moon phases (e.g., Easter) or ad-hoc decree cannot be easily expressed in a GTS form.

These symbolic abbreviations can in turn be used in GTS expressions to construct other schedules. For example, one can say "`JHCHRXME H08..12`" to indicate that the office hours on Christmas Eve is from 8 AM to 1PM only. And one can say "`JHNUSMEM..JHNUSLBR`" for the typical mid-western swimming pool season from Memorial Day to Labor Day.

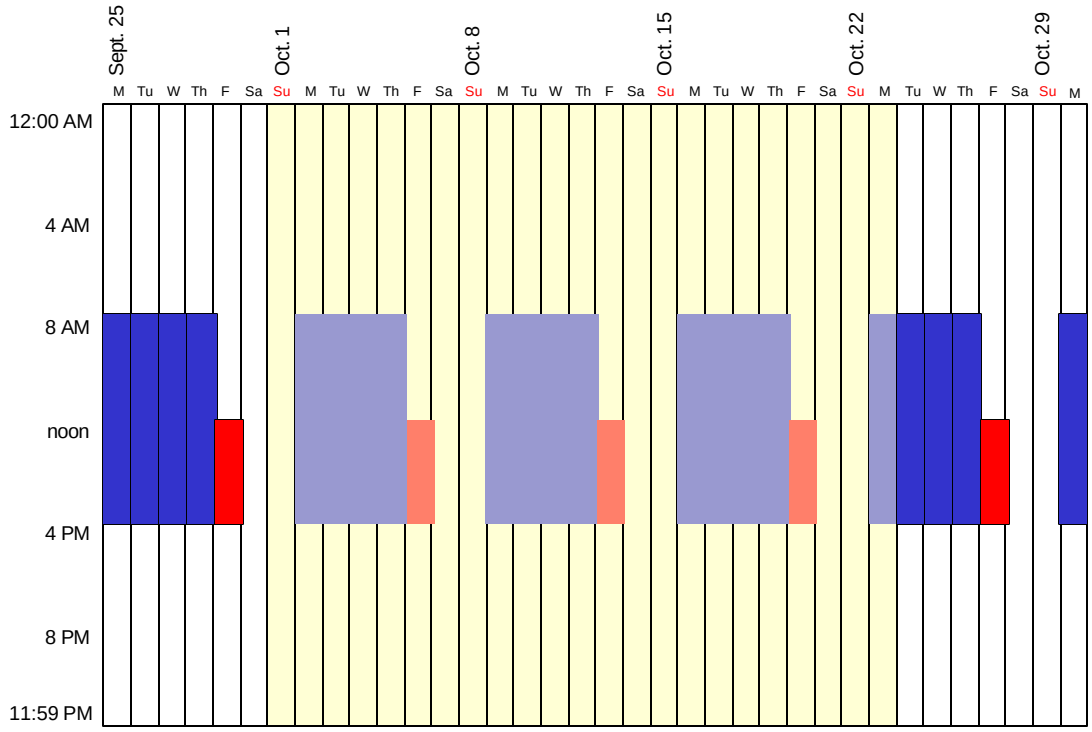

Figure 6: GTS Data Type Example

